**AI STOCK MARKET PREDICTION SOURCE CODE**

**Step 1: Set Up the Environment**

**Open a new notebook in Google Colab and ensure that the necessary libraries are installed. Google Colab typically comes with most required libraries pre-installed, but you can install any missing ones using pip.**

**python**

**CopyEdit**

**# Install yfinance if not already installed**

**!pip install yfinance**

**Step 2: Import Necessary Libraries**

**python**

**CopyEdit**

**import numpy as np**

**import pandas as pd**

**import yfinance as yf**

**import matplotlib.pyplot as plt**

**from sklearn.preprocessing import MinMaxScaler**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import LSTM, Dense**

**Step 3: Fetch Historical Stock Data**

**Replace 'XYZ' with the actual ticker symbol of Company XYZ.**

**python**

**CopyEdit**

**# Define the ticker symbol**

**ticker = 'XYZ' # Replace with actual ticker symbol**

**# Fetch historical data**

**data = yf.download(ticker, start='2015-01-01', end='2025-05-07')**

**closing\_prices = data['Close'].values.reshape(-1, 1)**

**Step 4: Normalize the Data**

**python**

**CopyEdit**

**scaler = MinMaxScaler(feature\_range=(0, 1))**

**scaled\_prices = scaler.fit\_transform(closing\_prices)**

**Step 5: Prepare the Dataset**

**python**

**CopyEdit**

**def create\_dataset(dataset, time\_step=60):**

**X, y = [], []**

**for i in range(len(dataset) - time\_step):**

**X.append(dataset[i:(i + time\_step), 0])**

**y.append(dataset[i + time\_step, 0])**

**return np.array(X), np.array(y)**

**time\_step = 60**

**X, y = create\_dataset(scaled\_prices, time\_step)**

**X = X.reshape(X.shape[0], X.shape[1], 1)**

**Step 6: Split into Training and Testing Sets**

**python**

**CopyEdit**

**train\_size = int(len(X) \* 0.8)**

**X\_train, X\_test = X[:train\_size], X[train\_size:]**

**y\_train, y\_test = y[:train\_size], y[train\_size:]**

**Step 7: Build the LSTM Model**

**python**

**CopyEdit**

**model = Sequential()**

**model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))**

**model.add(LSTM(50))**

**model.add(Dense(1))**

**model.compile(optimizer='adam', loss='mean\_squared\_error')**

**Step 8: Train the Model**

**python**

**CopyEdit**

**model.fit(X\_train, y\_train, epochs=20, batch\_size=32, verbose=1)**

**Step 9: Make Predictions**

**python**

**CopyEdit**

**# Predict the next day's closing price**

**last\_60\_days = scaled\_prices[-time\_step:]**

**last\_60\_days = last\_60\_days.reshape(1, time\_step, 1)**

**predicted\_price = model.predict(last\_60\_days)**

**predicted\_price = scaler.inverse\_transform(predicted\_price)**

**print(f"Predicted Closing Price for Next Day: {predicted\_price[0][0]:.2f}")**

**📊 Visualizing Model Performance**

**To evaluate the model's performance, you can plot the actual vs. predicted prices:**

**python**

**CopyEdit**

**# Predict on test data**

**predictions = model.predict(X\_test)**

**predictions = scaler.inverse\_transform(predictions)**

**actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))**

**# Plotting**

**plt.figure(figsize=(14,5))**

**plt.plot(actual, label='Actual Price')**

**plt.plot(predictions, label='Predicted Price')**

**plt.title(f'{ticker} Stock Price Prediction')**

**plt.xlabel('Time')**

**plt.ylabel('Price')**

**plt.legend()**

**plt.show()**

**OUTPUT:**

